## REMARKS/ARGUMENTS

This paper is being provided in response to the January 25, 2006 Final Office Action for the above-referenced application. In this response, Applicant has amended Claims 1, 30, 39, and 68 in order to clarify that which Applicant deems to be the claimed invention. Applicant respectfully submits that the amendments to the claims are all supported by the originally filed application.

Applicant gratefully acknowledges the indication of allowability of Claims 4, 6, 16-19, 25, 29, 31, 32, 34-38, 42, 44 54-57, 63, 67, and 69-76. Applicant notes that the Office Action indicates that Claims 34 and 38-42 are allowed. In light of the current rejections in the Office Action and the claim dependencies, Applicant believes that this is a typographical error and that Claim 34-38 and 42, rather than 34 and 38-42, are allowed.

The rejection of Claims 1-3, 5, 7-15, 20, 21, 39-41, 43, 45-53, 58 and 59 under 35 U.S.C. § 102(b) as being anticipated by Jas (U.S. Patent Publication No. 2002/0059260A1, hereinafter referred to as "Jas") is hereby traversed and reconsideration thereof is respectfully requested. Applicant respectfully submits that Claims 1-3, 5, 7-15, 20, 21, 39-41, 43, 45-53, 58 and 59, as amended herein, are patentable over the cited reference.

Applicant's Claim 1, as amended herein, recites a method for detecting an inconsistent data structure comprising: receiving a specification describing at least one consistency constraint of a data structure; and dynamically determining during execution of a program whether said data structure of said program violates said at least one consistency constraint, wherein said program is configured to perform said dynamically determining as part of consistency checking

processing in accordance with one or more of: a signal handler that processes a detected fault, wherein, upon the occurrence of said detected fault, said signal handler is invoked and wherein said signal handler subsequently invokes said consistency checking processing, a consistency check of a portion of said data structure at a first execution point in accordance with either a previous usage of said portion or a subsequent usage of said portion within said program, or at least one user specified execution point of said program. Claims 2, 3, 5, 7-15, 20 and 21 depend from Claim 1.

Applicant's Claim 39, as amended herein, recites a computer program product that detects an inconsistent data structure, the computer program product comprising executable code stored thereon for execution by a processor that: receives a specification describing at least one consistency constraint of a data structure; and dynamically determines during execution of a program whether said data structure of said program violates said at least one consistency constraint, wherein said program is configured to perform said dynamically determining as part of consistency checking processing in accordance with one or more of: a signal handler that processes a detected fault, wherein, upon the occurrence of said detected fault, said signal handler is invoked and wherein said signal handler subsequently invokes said consistency checking processing, a consistency check of a portion of said data structure at a first execution point in accordance with either a previous usage of said portion or a subsequent usage of said portion within said program, or at least one user specified execution point of said program. Claims 40, 41, 45-53, 58 and 59 depend from Claim 39.

Jas discloses a computer-based software method for creating and modifying a database which contains a multiplicity of records consisting of attribute records, entity records, and

constraint record. (See Abstract; Par. 56-62). Jas's Figure 1 includes an entity manager 102 and an entity validator 105. The entity manager 102 may present the entity validator 105 with each new and modified entity to check for consistency with the constraints before storing the new or changed entity. (Figure 1, Par. 81).

Applicant's Claim 1, as amended herein, is neither disclosed nor suggested by Jas in that Jas neither discloses nor suggests at least the features of ***a method for detecting an inconsistent data structure comprising: ... dynamically determining during execution of a program whether said data structure of said program violates said at least one consistency constraint, wherein said program is configured to perform said dynamically determining as part of consistency checking processing in accordance with one or more of: a signal handler that processes a detected fault, wherein, upon the occurrence of said detected fault, said signal handler is invoked and wherein said signal handler subsequently invokes said consistency checking, processing a consistency check of a portion of said data structure at a first execution point in accordance with either a previous usage of said portion or a subsequent usage of said portion within said program, or at least one user specified execution point of said program,*** as set forth in Claim 1.

The feature of the "signal handler" as set forth in amended Claim 1 is described, for example, beginning at page 28 line 5 of Applicant's specification which states that a program may catch signals, such as runtime exceptions and other programming conditions including, but not limited to, divide by zero errors, segmentation fault violations, and the like. Such faults may be caused by inconsistent data structures causing a signal handler to be invoked which may further invoke the consistency checker and repairer.

As support for disclosing the above-referenced "signal handler" feature of Claim 1, the Office Action cites paragraphs 85, and 87-89 of Jas which describe elements of Jas including the constraint manager 103, the entity manager 102 and the entity validator 105. In particular, paragraph 85 of Jas relates to the entity validator 105 evaluating the antecedent statement for constraints managed by the constraint manager 103 as described in paragraph 84. If the antecedent statement evaluates to TRUE, then the consequent statement is evaluated. If the consequent statement evaluates to FALSE, then the entity under consideration is considered invalid. The result is returned to the entity manager 102 by the entity validator 105.

As pointed out above, paragraph 81 of Jas discloses performing consistency checking with constraints before storing each new or changed entity. The entity manager 102 presents the entity validator 105 with each new and modified entity to check for consistency with constraints before storing the new or changed entity. For each attribute, the entity validator 105 retrieves the attribute description record from the attribute description manager 104. Paragraph 85 of Jas discloses subsequent steps in this process of performing consistency checking for each new or changed entity evaluated by the entity evaluator 105. Applicant is unclear as to whether the Office Action is considering the entity validator 105 or the entity manager 102 as the "signal handler" recited in Applicant's Claim 1. In either case, Applicant respectfully submits that Jas neither discloses nor suggests at least the foregoing recited feature of Applicant's amended Claim 1. Jas discloses an entity manager 102 invoking an entity validator 105 for each new and modified entity. The entity validator 105 determines whether each such entity is consistent using constraints and reports any invalid entities to the entity manager 102. Jas neither discloses nor suggests ***a signal handler that is invoked upon the occurrence of a detected fault,*** and a ***signal***

***handler that subsequently invokes said consistency checking,*** as set forth in amended Claim 1. If the entity manager 102 is the "signal handler", the entity manager 102 is not invoked upon the occurrence of a detected fault and the entity manager 102 does not subsequently invoke consistency checking processing, as recited in Claim 1. Rather, the entity manager 102 is invoked for each new and modified entity and invokes the entity validator 105 to perform consistency checking for each new and modified entity. If the entity validator 105 is the "signal handler", the entity validator 105 is not invoked upon the occurrence of a detected fault and the entry validator 105 does not further invoke consistency checking processing upon the occurrence of a detected fault, as recited in Claim 1. Rather, the entry validator 105 is invoked to perform consistency checking for each new and modified entity.

Applicant notes that the Office Action does not set forth any citations in Jas as support for disclosing ... ***wherein said program is configured to perform said dynamically determining as part of consistency checking processing in accordance with one or more of: ... a consistency check of a portion of said data structure at a first execution point in accordance with either a previous usage of said portion or a subsequent usage of said portion within said program, or at least one user specified execution point of said program,*** as set forth in Claim 1. Applicant respectfully submits that Jas appears silent regarding any disclosure or suggestion of these features of Claim 1.

For at least these reasons, Applicant respectfully submits that Claim 1, as amended herein, is neither disclosed nor suggested by Jas.

Applicant respectfully submits that, although claims that depend from independent Claim 1 are patentable for at least the reasons set forth above regarding Claim 1, claims that depend from Claim 1 set forth additional patentable features. For example, Claim 13 recites ***wherein said description of said abstract model includes at least one model definition rule and at least one declaration for one of: a set and a relation, said at least one model definition rule representing an element of said data structure in at least one of a set and a relation.*** Applicant respectfully submits that Jas, as a database system, does not disclose or suggest representing an element of a data structure from a program using model definition rules of an abstract model. Model definition rules may be used to represent data structures of a program in an abstract model. Rather, Jas discloses in paragraphs 55-56 using a model including an attribute description, a constraint and an entity, not a model definition rule. The entity is the data in the database, not a data structure of a program.

For reasons similar to those set forth regarding Claim 1, Claim 39 is also neither disclosed nor suggested by Jas in that Jas neither discloses nor suggests at least the features of ***a computer program product that detects an inconsistent data structure, the computer program product comprising executable code stored thereon for execution by a processor that: ... dynamically determines during execution of a program whether said data structure of said program violates said at least one consistency constraint, wherein said program is configured to perform said dynamically determining as part of consistency checking processing in accordance with one or more of: a signal handler that processes a detected fault, wherein, upon the occurrence of said detected fault, said signal handler is invoked and wherein said signal handler subsequently invokes said consistency checking processing, a consistency check of a portion of said data structure at a first execution point in accordance with either a***

***previous usage of said portion or a subsequent usage of said portion within said program, or at least one user specified execution point of said program,*** as set forth in Claim 39.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

The rejection of Claims 22-24, 26-28, 60-62, and 64-66 under 35 U.S.C. § 102(e) as being anticipated by Applin (U.S. Patent Publication No. 2004/0015876A1, hereinafter referred to as "Applin") is hereby traversed and reconsideration thereof is respectfully requested. Applicant respectfully submits that Claims 22-24, 26-28, 60-62, and 64-66 are patentable over the cited reference.

Claim 22 recites a method of dynamically repairing an inconsistent data structure during program execution comprising: receiving at least one inconsistency violation; selecting a repair to correct said at least one inconsistency violation; and repairing said inconsistent data structure, said repairing including modifying at least a portion of said inconsistent data structure. Claims 23, 24, and 26-28 depend from Claim 22.

Claim 60 recites a computer program product that dynamically repairs an inconsistent data structure during program execution, the computer program product comprising executable code stored thereon for execution by a processor that: receives at least one inconsistency violation; selects a repair to correct said at least one inconsistency violation; and repairs said inconsistent data structure, said repairs including modifying at least a portion of said inconsistent data structure. Claims 61, 62, and 64-66 depend from Claim 60.

Applin discloses performing checks to verify proper pointer usage and initiate any appropriate processing to avoid a fatal error, execution of an invalid operation and/or provide notification of an error, warning, or other condition. Applin discloses checking for a pointer not properly aligned for the type of data to be retrieved, and other pointer usage inconsistent with a particular operation, data type, etc. Applin discloses processing if a misuse or potential misuse is detected including, for example, terminating program execution and generating an error message, providing a warning message without terminating program execution, providing alternative processing and/or default values in response to identification of a pointer problem without program termination. Applin discloses determining and initiating a desired action. Applin discloses a safe or recovery state being identified and instantiated when an inappropriate use of the null pointer is attempted. (Par. 15-16; Figure 1).

Applicant's Claim 22 is neither disclosed nor suggested by Applin in that Applin neither discloses nor suggests at least the features of ***a method of dynamically repairing an inconsistent data structure during program execution comprising: ... repairing said inconsistent data structure, said repairing including modifying at least a portion of said inconsistent data structure,*** as set forth in Claim 22.

As support for disclosing the foregoing recited feature of Claim 22, the Office Action at page 5 cites paragraphs 15 and 16 of Applin. The Office Action on page 5 indicates that paragraph 15 of Applin discloses providing a default value which is relied upon as support for disclosing ***said repairing including modifying at least a portion of said inconsistent data structure,*** as set forth in Claim 22. The Office Action appears to state that the disclosure of

providing a default value in Applin means that the inconsistent data structure is modified. Applicant respectfully submits that providing a default value neither discloses nor fairly suggests that the inconsistent data structure is modified. A default value may be supplied, for example, such as a fixed value which is returned whenever a particular fault is detected. For example, if a null pointer is detected in connection with a read operation, a default value may be returned since the null pointer cannot be used to read a value. There is no modification of the null pointer detected as inconsistent. Rather than perform a modification, a default value is returned as a result of the read operation. Applin discloses, in response to pointer misuse, performing processing including supplying a default value and identifying a safe or recovery state. However, Applin does not further disclose or suggest that the operation of providing a default value or identifying a safe or recovery state includes modifying the inconsistent data structure. Accordingly, Applin neither teaches, discloses nor suggests at least the foregoing recited features of Claim 22.

For reasons similar to those set forth regarding Claim 22, Applicant's Claim 60, as amended herein, is neither disclosed nor suggested by Applin in that Applin neither discloses nor suggests at least the features of ***a computer program product that dynamically repairs an inconsistent data structure during program execution, the computer program product comprising executable code stored thereon for execution by a processor that: ... repairs said inconsistent data structure, said repairs including modifying at least a portion of said inconsistent data structure,*** as set forth in Claim 60.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

The rejection of Claims 30, 33, 77, 78, 68, 71, 79, and 80 under 35 U.S.C. 102(e) as being anticipated by Burrows (US Patent Publication US2003/0125290 A1, hereinafter "Burrows") is hereby traversed and reconsideration thereof is respectfully requested. Applicant respectfully submits that Claims 30, 33, 77, 78, 68, 71, 79, and 80, as amended herein, are patentable over the cited reference.

Claim 30, as amended herein, recites a method of handling an invalid memory reference comprising: determining whether a memory reference associated with an operation is invalid; and if said memory reference is invalid, performing an action selected in accordance with a type of said operation, wherein said type includes at least one of a read operation or a write operation, wherein, when said action allows a program including said operation to continue executing, said action is a substitute action performed in place of said operation. Claims 30, 77 and 78 depend from Claim 30.

Claim 68, as amended herein, recites a computer program product that handles an invalid memory reference comprising executable code stored thereon for execution by a processor that: determines whether a memory reference associated with an operation is invalid; and if said memory reference is invalid, performs an action selected in accordance with a type of said operation , wherein said type includes at least one of a read operation or a write operation, wherein, when said action allows a program including said operation to continue executing, said action is a substitute action performed in place of said operation. Claims 71, 79 and 80 depend from Claim 68.

Burrows relates generally to checking memory validity, and more specifically, to dynamically determining memory errors through data-type checking. (Par. 1). Burrows' Figure 5 discloses an embodiment that uses a program interpreter. (Par 48; Figure 5). At step 504, a determination is made if execution of the instruction is inconsistent with shadow array information. If the interpreter determines that the execution of the instruction is inconsistent with the shadow array information, it generates a report of the error at step 506. (Par. 49) From step 506, for fatal errors or user requested termination, control proceeds to step 512 (See Figure 5). An error may also generate an interrupt message which then causes the interpreter to inform the user and to query the user. In another embodiment, the interpreter may return directly to a debugging program or other interface directly without querying the user. If the interpreter determines that execution of the instruction is consistent with the shadow array information, or if the interpreter determines that it is inconsistent but the error is not fatal, or the user does not require termination of the program, the interpreter executes the instruction at step 508. (Par. 50, 51).

Applicant's Claim 30, as amended herein, is neither disclosed nor suggested by Burrows in that Burrows neither discloses nor suggests at least the features of ***a method of handling an invalid memory reference comprising: ... if said memory reference is invalid, performing an action selected in accordance with a type of said operation, wherein said type includes at least one of a read operation or a write operation, wherein, when said action allows a program including said operation to continue executing, said action is a substitute action performed in place of said operation,*** as set forth in Claim 30. As discussed above with reference to Figure 5 of Burrows, Burrows discloses determining at step 504 if execution of the instruction is inconsistent with the shadow array information. From step 504, control proceeds to step 506 to

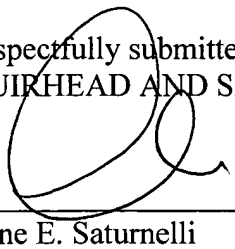
generate a report. If there has been a fatal error or a user requested termination, control proceeds to step 512. Otherwise, control proceeds to step 508 to execute the instruction causing the inconsistency. Thus, Burrows discloses that if the program continues execution upon detection of an inconsistency, control proceeds to step 508 where the instruction determined as inconsistent is executed. This is in contrast to Applicant's Claim 30 which recites ***wherein, when said action allows a program including said operation to continue executing, said action is a substitute action performed in place of said operation***. Burrows discloses performing the instruction determined as inconsistent rather than performing an action in place of the operation associated with the invalid memory reference, as set forth in Applicant's amended Claim 30.

For reasons similar to those set forth regarding Claim 30, Applicant's Claim 68 is neither disclosed nor suggested by Burrows in that Burrows neither discloses nor suggests ***a computer program product that handles an invalid memory reference comprising executable code stored thereon for execution by a processor that: ... if said memory reference is invalid, performs an action selected in accordance with a type of said operation , wherein said type includes at least one of a read operation or a write operation, wherein, when said action allows a program including said operation to continue executing, said action is a substitute action performed in place of said operation***, as set forth in Claim 68.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

Based on the above, Applicant respectfully requests that the Examiner reconsider and withdraw all outstanding rejections and objections. Favorable consideration and allowance are earnestly solicited. Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at 508-898-8604.

Respectfully submitted,
MUIRHEAD AND SATURNELLI, LLC



Anne E. Saturnelli
Reg. No. 41,290

Muirhead and Saturnelli, LLC
200 Friberg Parkway, Suite 1001
Westborough, MA 01581
Tel: (508) 898-8601
Fax: (508) 898-8602

Date: April 20, 2006